Toward a correct and optimal time-aware cloud resource allocation to business processes

Rania Ben Halima, Slim Kallel, Walid Gaaloul, Zakaria Maamar, Mohamed Jmaiel

Toward a Correct and Optimal Time-aware Cloud Resource Allocation to Business Processes

Rania Ben Halima[a,b], Slim Kallel[b], Walid Gaaloul[a], Zakaria Maamar[c], Mohamed Jmaiel[b,d]

[a] *Telecom SudParis, UMR 5157 Samovar, Institut Polytechnique de Paris, France*
[b] *ReDCAD, University of Sfax, Tunisia*
[c] *Zayed University, Dubai, United Arab Emirates*
[d] *Digital Research Center of Sfax, Technopole Sfax, Tunisia*

Abstract

Cloud is an increasingly popular computing paradigm that provides on-demand services to organizations for deploying their business processes over the Internet as it reduces their needs to plan ahead for provisioning resources. Cloud providers offer competitive pricing strategies (e.g., on-demand, reserved, and spot) specified based on temporal constraints to accommodate organizations' changing and last-minute demands. Despite their varieties and benefits to optimize business process deployment cost, using those pricing strategies can lead to violating time constraints and exceeding budget constraints due to inappropriate decisions when allocating cloud resources to business processes. In this paper, we present an approach to guarantee a correct and optimal time-aware allocation of cloud resources to business processes. Correct because time constraints on these processes are not violated. And, optimal because the deployment cost of these processes is minimized. For this purpose, our approach uses timed automata to formally verify the matching between business processes' temporal constraints and cloud resources' time availabilities and linear programming to optimize deployment costs. Experiments demonstrate the technical doability of our proposed approach.

Keywords: Business process, Cloud resource, Formal verification, Optimization

1. Introduction

Cloud computing is an attractive operational model for organizations that wish, among other reasons, to reduce upfront investment on Information and Communication Technologies (ICT) and to tap into hardware and software resources of cloud providers in return of a fee. Cloud is known for resource elasticity and pay-per-use model making it perfect for organizations that witness a surge of activities during particular periods of the year. For instance, during 2017 Christmas Amazon.com had to temporarily cope with 280 millions online retail transactions calling for immediate provisioning of resources that luckily were released once the load went back to normal[1].

In today's economic world, attracting and retaining customers constitutes a challenge. Indeed, many cloud providers offer competitive pricing strategies (e.g., on-demand, reserved, and spot) to accommodate users' changing and last-minute demands. However this price variation puts more pressure on cloud providers who need to ensure resource availability on a short-notice, for example. Organizations that "wrestle" with time like shipping need to respond quickly to any unforeseen event and hence, could call upon cloud providers anytime. Indeed, cloud computing allows organizations to optimize their Business Processes (BPs, *aka* know-how) thanks to different techniques associated with cloud computing such as virtualization and load balancing. But, this optimization should not happen on the expense of increasing operation costs [1] and/or violating time constraints, for example. "Striking" the right balance between cloud resources' pricing strategies and BPs' time constraints is one of our objectives in this paper. We achieve this objective by ensuring the temporal correctness of cloud-aware BPs, finding an optimal-deployment cost for these BPs, and validating the deployment of these BPs as well.

Some research works on temporal verification of BPs [2, 3] and allocation of cloud resources to BPs [1, 4] are reported in the literature. Nevertheless,

[1]www.hitwise.com.

formal satisfaction of BPs' temporal constraints with respect to cloud resources' availabilities and identification of optimal deployment cost of BPs over these resources, is either barely touched upon or handled on a case-by-case basis. Our previous work reported in [5] and [6] is one step towards a formal specification and verification of allocating cloud resources to BPs. To this end we used timed-automata networks to check BPs' time-constraint behaviors like reachability and deadlock-free [7]. We also used linear programming to optimize the deployment cost of these BPs [8]. Although, due to the complexity of linear programming, handling BPs with large number of activities (200 as per our work in [8]) turned out cumbersome and inefficient. In this paper, first, we extend our verification approach presented in [5, 6] to check more advanced properties such as liveness to guarantee better correctness of time-constrained, cloud-aware BPs. Second, we improve our optimization approach presented in [8] to reduce the deployment cost of cloud resources when running more complex and "large" BPs.

Our contributions are, but not limited to, (i) developing a set of rules to transform BP into timed-automata as a step towards a correct time-aware cloud resource allocation in BP, (ii) formalizing the optimization problem as a mathematical model to minimize the deployment cost of time-constrained BPs, and (iii) evaluating the technical doability of our approach for ensuring the correctness and optimization of time-aware cloud resource allocation to BPs.

The remainder of this paper is organized as follows: Section 2 presents the necessary concepts related to our work. Section 3 presents a real use case from France Telecom Orange labs. Section 4 and Section 5 discuss the verification and optimization steps, respectively. Section 6 details the evaluation. Section 7 presents some related works. Finally, Section 8 presents our conclusions and future works.

2. Preliminaries

In this section, we present the foundations upon which our approach for the correctness and optimization of BPs over cloud resources, is built. First, we define cloud resources along with their pricing strategies and then, present types of temporal constraints that BPs' activities could be subject to. Finally, we present timed automata elements.

2.1. Cloud resource

Cloud computing is known for ∗aaS model with focus here on computing resources of type Infrastructure-as-a-Service (IaaS).

Definition 1. *A cloud resource is a couple (id, Cap) where:*

- *id is a unique identifier;*
- *Cap is a resource' capacity in terms of memory amount and virtual core number, Cap=(RAM, vCPU)* $\in \mathbb{N} \times \mathbb{N}$.

Cloud resources payment refers to different pricing strategies that are vendor dependent such as *pre-paid subscription* for Microsoft [9], *per-minute billing* for Google [10], and *on-demand, reserved*, and *spot* for Amazon. Hereafter we discuss Amazon's 3 types of pricing [11]; they are comprehensive and cater to major users' needs in terms of cloud resources.

- *On-demand:* the customer pays an hourly fixed amount with no long-term commitment. The procured resource capacity can be increased or decreased depending on the customer applications' requirements and the payment is done for the procured capacity, only.
- *Reserved:* the customer can make a one-time, all upfront, partial upfront, or no upfront payment for a long-term commitment of a resource capacity and pays a significant hourly rate when running capacity instances in the future.
- *Spot:* customers bid for unused resource capacities to secure some spots instances offered at a spot price and with an interruption risk due to the bidding process. In response to potential disruptions, Amazon also proposes spot instances with a predefined duration (also known as *Spot Blocks*) that are continuously available (from 1 to 6 hours).

Definition 2. *A cloud resource pricing strategy is a triple st=(type,TC,c) where:*

- *type is a type of strategy;*
- *TC is the temporal availability of a price that the strategy st imposes;*
- *c is a unit hour price that the strategy st proposes, $c \in \mathbb{R}$.*

Temporal constraints over a cloud resource's pricing strategy are of 2 types:

- Relative temporal constraints specify a time interval $[MinAvr, MaxAvr]$ in which a resource is available at a certain price; $1 \leq MinAvr \leq MaxAvr$.
- Absolute temporal constraints specify the start and finish times of a resource availability at a certain price. Those constraints could be specialized into: Start Using No Earlier Than ($SUNET(r)$), Finish Using No Earlier Than ($FUNET(r)$), Start Using No Later Than ($SUNLT(r)$), and Finish Using No Later Than ($FUNLT(r)$).

Let's consider a set of n cloud resources that p cloud providers propose in s pricing strategies:

- $\mathscr{R} = \{r_i \mid 1 \leq i \leq n\}$ is the set of all cloud resources;
- $Pr_i = \{pr_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq p\}$ is the set of all providers offering a cloud resource r_i;
- $St_{ij} = \{st_{ijk} \mid 1 \leq i \leq n, 1 \leq j \leq p, 1 \leq k \leq s\}$ is the set of all pricing strategies for each pr_{ij}.

Table 1 presents a set of Amazon cloud resources. Their operating system is Linux and availability time zone is us-east-a1. In our current work, we assume that the size of transferred data between cloud resources is not considered and is left as future work.

Amazon EC2 offers different instance types with different computing capacities (virtual CPU cores ($vCPU$) and memory (RAM)) and grouped into instance families. For instance, $r_3 = m3.2xlarge$ corresponds to an Amazon computing

Table 1: Virtual machine instance properties by pr_{i1}=Amazon EC2

Instances	**RAM (GB)**	**vCPU**	**On-demand**	**Reserved** (no-upfront)	**Spot blocks**	**Spot instance**
r_1= m4.xlarge	16	4	c_{111}=0.215\$/h	c_{112}=0.147\$/h	c_{113}=0.129\$/h [0h,1h]	c_{114}=0.0491\$/h [6pm,1am$^{(+1)}$]
					c_{113}=0.142\$/h [1h,6h]	c_{114}=0.0386\$/h [1am,6pm]
r_2= r3.large	15	2	c_{211}=0.166\$/h	c_{212}=0.105\$/h	c_{213}=0.096\$/h [0h,1h]	c_{214} =0.0225\$/h [3am,10pm]
					c_{213}=0.102\$/h [1h,6h]	c_{214}=0.0381\$/h [10pm,3am$^{(+1)}$]
r_3 = m3.2xlarge	30	8	c_{311}=0.532\$/h	c_{312}=0.380\$/h	c_{313}=0.293\$/h [0h,1h]	c_{314}=0.0787\$/h [10am,9pm]
					c_{313}=0.372\$/h [1h,6h]	c_{314}=0.0863\$/h [9pm,10am$^{(+1)}$]

resource of type *m3*. The latter belongs to *general purpose* instance family that provides a balance of compute and memory resources, and can be used for a variety of workloads [11]. Still in Table 1, the unit hourly price of r_3 as spot instance is equal to c_{313}=0.372\$/h. As a result, r_3 is temporally available for a minimum duration $MinAvr_3$ of 1 hour and a maximum duration $MaxAvr_3$ of 6 hours (i.e., st_{313}=(spot, [1h,6h], 0.372\$/h)).

2.2. Business process

As per Definition 3, a BP, consists of a set of activities A (where $A = \{a_q : q \in \{1, \cdots, z\}\}$) that are performed in coordination in an organizational and technical environment to jointly achieve business goals [12]. It happens that a BP is subject to some time constraints which can be relative or absolute. For more details about temporal constraints, we refer readers to [13].

- Relative temporal constraints refer to activity duration and dependency. Duration defines the completion time of an activity expressd as an interval $[MinD_{a_q}, MaxD_{a_q}]$. Let s(a) (resp. e(a)) be the starting (resp. the ending) time of an activity a. Let $MinD_{a_q}$ and $MaxD_{a_q}$ be two relative time values representing respectively the minimum and maximum durations of a. Duration is defined as follow:

$$\textit{Duration}(a, MinD_{a_q}, MaxD_{a_q}) \overset{\text{def}}{=} MinD_{a_q} \leq \text{e}(a) - \text{s}(a) \leq MaxD_{a_q}$$

Dependency is a relationship between two activities, a_q and a_l ($l \neq q$), in which one activity depends on the start or finish of another in order to begin or end [13]. We consider 4 temporal dependencies as follow; *Start-To-Finish (SF), Start-To-Start (SS), Finish-to-Start (FS), and Finish-to-Finish (FF).* For illustration, Finish-to-Start dependency between two activities a_q and a_l, is defined as follows:

$$\textit{TD}(\text{FS}, a_q, a_l, D_{min}, D_{max}) \overset{\text{def}}{=} D_{min} \leq \text{s}(a_l) - \text{e}(a_q) \leq D_{max}$$

This definition denotes that a_l should start its execution no later than D_{max} time units and no earlier than D_{min} time units after a_q ends.

- Absolute temporal constraints define a punctual temporal structure and refer to start and finish times of an activity. First, Must Start On (MSO) and Must Finish On (MFO) indicate the exact time for begining and completing an activity. Second, Start No Earlier Than (SNET) and Finish No Earlier Than (FNET) indicate the earliest possible time that an activity can begin and complete. Finally, Start No Later Than (SNLT) and Finish No Later Than (FNLT) indicate the latest possible time that an activity is set to begin and complete.

In previous work [14], we proposed a survey of time-related aspects of process models. We identified the existing approaches that specify and verify temporal constraints and enhance the time dimension in the BPM field. Some approaches opted for modeling time using graphs while others used formal specification languages and algebras (e.g., Linear Temporal Logic [15], Allen's algebra [16], and time petri nets [17]) for specifying and verifying time-based BP models. In the following, we compare our time representation to Allen's algebra. When using Allen's algebra, we identified several limitations since the designer can not specify that:

(i) A BP activity should be executed with respect to a specific time nor a date.

(ii) A BP activity can be executed a limited number of times within a time interval.

(iii) A set of BP activities should be performed periodically in a predefined duration.

Contrarily, we examined in our research work [13, 18] how to extend BPMN artifacts to specify expressive temporal constraints. Compared to Allen's algebra, we can represent:

(i) Absolute temporal constraints: MSO/MFO (Must Start/Finish On a defined date), SNLT/FNLT (Start/Finish No Earlier Than). These constraints mean that a BP activity should be executed in a defined date.

(ii) Temporal constraints over cardinality that specify that a BP activity should be executed n times in a specific interval.

(iii) Periodic/Sporadic activity: A periodic activity has a recurrent start event every T period of time. T is the time duration between the occurrences of two start events of the same activity. While a sporadic activity has a recurrent start event, but the period is variant. The sporadic activity has a minimum and maximum separation between the occurrence of two start events in the same activity which called sporadic interval.

For the sake of illustration, we only consider in this paper, absolute temporal constraints, duration constraints, and temporal dependencies between activities. To this end, we relied on (i) BPMN to specify and model a BP that is subject to some time constraints that could be either relative or absolute and (ii) timed automata to formally verify the correctness of a time-constrained cloud resources allocation to BPs.

Definition 3. *A BP model is a tuple (N, E, F, Req_A, Pen_A) where:*

- *N is the set of nodes that correspond to a set of activities A, gateways G, and events Ev, i.e., N=$\{A \cup G \cup Ev\}$;*

- *$E \subseteq N \times N$ is the set of edges;*
- *$F : A \longrightarrow T$ is a function that assigns temporal constraints T to activities A;*
- *Req_A is the set of activities requesting cloud resources;*
- *Pen_A is the set of financial penalties that are subject to in case activities are canceled due to resource interruption.*

Definition 4. *An activity a is a tuple (temp, res, type) where:*

- *temp is a set of temporal constraints which can be temporal duration, dependency, and/or absolute temporal constraints;*
- *res is the resource allocated;*
- *type is the pricing strategy type.*

2.3. Timed automata

As per Definition 5, a timed automata is a directed graph where nodes correspond to the system states (*aka* locations) and edges correspond to transitions between these nodes. In a timed automata [19], the time passing is modeled by real non-negative variables, named clocks. The conditions over clocks are defined through the use of constraints allowing to reset certain clocks to zero. A clock constraint, called invariant, is associated with each state. While this invariant is satisfied, the system remains in the current state. The transitions are labelled by both temporal constraints, called guards, and clock variables, and are synchronized through binary channels. Guards and invariants are conjunctions of constraints $x \bowtie v$, where x is in the set of non-negative clocks X, $v \in \mathbb{R}^+$, and $\bowtie \in \{<, \leq, =, >, \geq\}$. A transition is enabled if the guard evaluates to true and the source state is active. The set of constraints over X is named $\Psi(X)$.

Definition 5. *A timed automata is a tuple (L, X, l_0, Tr, I) where:*

- *L is a finite set of states,*
- *X is a finite set of clocks,*
- *l_0 is an initial state,*
- *Tr $\sqsubseteq L\times\ \Psi(X)\times 2^X\times\ L$ is the set of transitions,*
- *I: L $\longrightarrow \Psi(X)$ is a function that assigns invariants to states.*

Definition 6. *A transition is a tuple tr=(l, α, ψ, cl, l') $\in$ E where l is a source state, l' is a target state, α is a label, ψ is a guard, and cl is a set of clocks to reset.*

3. Case study

For illustration purposes, we consider "supervision service" BP from France Telecom/Orange labs ([20], Figure 1). First, we model this BP's process model in Business Process Model Notation (BPMN) [21], the standard for BP modeling, and then specify cloud resources and their pricing strategies, and activities' temporal constraints.

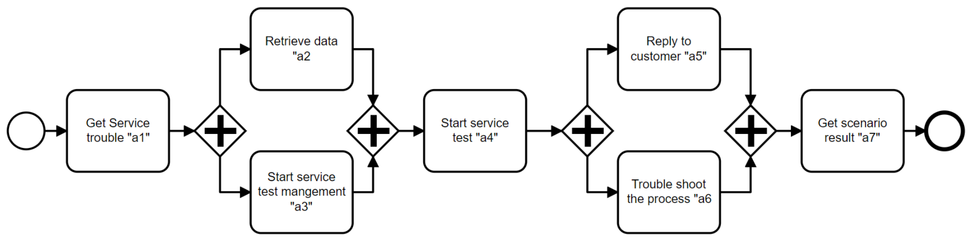


Figure 1: Supervision service BP in BPMN

Table 2 lists "supervision service" BP's temporal constraints along with the capacities expressed in terms of *RAM* and *vCPU* that each activity in this process requires. For instance, a_2 and a_6 minimum duration is 2 h and maximum duration is 3 h. Moreover, the time lag between a_3 end and a_4 start should be between 3 h and 5 h. a_1 and a_5 require a resource with 16 GB of RAM and 4 virtual CPUs. Moreover, some activities are subject to penalty cancellation prices p_q that should be added to the BP's deployment cost. For instance, a_1 has a penalty price p_1=0.7$ (Table 2). The BP is triggered when a customer sends a compliant by executing get service trouble ticket activity a_1. Then, necessary data are retrieved a_2 so that the management test begins a_3. After launching service test a_4 the process continues by replying to the customer a_5 and troubleshooting a_6. The process ends when sending results to the customer (get test scenario results a_7).

Table 2: Process activities' temporal constraints and needs in cloud resources

Activities	a_1	a_2	a_3	a_4	a_5	a_6	a_7
Durations	[1h,2h]	[2h,3h]	[1h,1h]	[1h,4h]	[1h,2h]	[2h,3h]	[1h,2h]
Penalties in $	0.7	0	0	0.2	0	0	0
RAM in GB	16	15	16	28	16	28	30
vCPU number	4	2	2	8	4	8	8

When working on "supervision service" BP, the designer has different options in terms of resource allocation. For instance, since r_1 satisfies a_1's RAM and vCPU requirements, then he assigns r_1 to a_1 as a spot instance though we will illustrate later that this assignment would violate a temporal constraint. Namely, r_1 is available from 1am until 6pm at a price of 0.0386$/h. However, if the process starts at 10pm then this would lead to a temporal constraint violation, i.e., 10pm $\notin [1am, 6pm]$. In conjunction with analyzing r_1, r_2 satisfies a_7's capacity requirements. The designer can consider r_2 as a spot blocks that would cover the minimum and maximum execution-duration of 1 hour at a price of 0.096$/h. The maximum duration of a_7 is 2 hours. Unfortunately, the designer's choice can lead again to a temporal violation; the temporal duration of the activity is greater than the duration of the selected resource. Besides, it happens that an activity can use one cloud resource that would has different prices according to the available pricing strategies. Table 3 presents 2 possible resource allocation options with focus on a_1. In the first allocation r_1 is allocated as a spot instance and in the second allocation as an on-demand instance. We compute the cost for each allocation based on Equation 1 that is (i) the sum of the unit hourly price c_{ijk} of r_i from pr_{ij} in strategy st_{ijk} by the activity duration d_q (we assume that $d_q = MaxD_{a_q}$) and (ii) the sum of activity penalty prices [8]. Although a more expensive strategy (on-demand) is used, the BP cost is lower for the second allocation. This is due to the penalty price of a_1 that is added to the process cost in case of allocating a spot instance with an interruption risk. Therefore, it is important to find the suitable cloud resource and the best pricing strategy for each cloud resource especially in case of critical activities. More precisely, it is better to avoid selecting spot strategy for cloud resources allocated for critical activities. Consequently, to minimize the BP cost deployment, the designer needs (i) to find an optimal solution that satisfies activities' requirements, and/or (ii) to reduce the search complexity of the optimization problem to converge in a short time to an optimal solution if the BP is large.

$$C = \sum_{q=1}^{|A|} \sum_{i=1}^{|R|} \sum_{j=1}^{p} \sum_{k=1}^{s} d_q c_{ijk} + \sum_{q=1}^{|A|} p_q \qquad (1)$$

4. Verification of cloud resources allocation correctness

We present, in this section, the necessary steps that we took to ensure the correctness of our time-constrained cloud resources allocation to BPs. First, we developed a set of rules to automatically transform time-constrained BPMN models into a network of timed automata. Then, we formally verified this network against advanced properties known in the community as liveness, deadlock free, and deadline.

Table 3: Possible resource allocations for supervision service BP

Activity	$Allocation_1$	$Allocation_2$
a_1	r_1 from pr_1 from st_{114}	r_1 from pr_1 from st_{111}
a_2	r_2 from pr_1 from st_{213}	r_2 from pr_1 from st_{213}
a_3	r_1 from pr_1 from st_{113}	r_1 from pr_1 from st_{113}
a_4	r_5[a] from pr_2 from st_{521}	r_5 from pr_2 from st_{521}
a_5	r_4 from pr_2 from st_{421}	r_4 from pr_2 from st_{421}
a_6	r_3 from pr_1 from st_{314}	r_3 from pr_1 from st_{314}
a_7	r_3 from pr_1 from st_{313}	r_3 from pr_1 from st_{313}
Total cost	3.7578$	3.4106$

[a] r_4 (c_{421}= 0.128$/h) and r_5 (c_{521}= 0.387$/h) are Microsoft Azure instances.

4.1. From BPMN to timed automata

We recall that our BPMN process model consists of activities that are subject to temporal constraints and consume cloud resources. We developed a set of transformation rules that take as an input a BP's activities along with the cloud resources they consume and produce as an output a network of timed automata depending on these resources' pricing strategies.

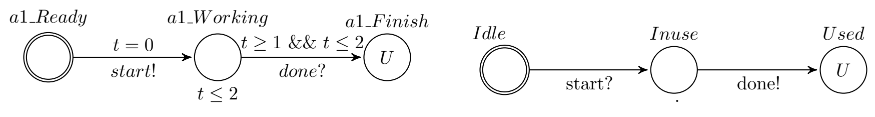
A network of timed automata consists of (i) the BP's timed automata with focus on its activities' states and temporal constraints and (ii) the cloud resources' set of timed automata that these activities will consume. The synchronisation of BP-related and resource-related timed automata is achieved using binary channels.

To begin with, let us consider an activity a that consumes a cloud resource r. Because of this resource's pricing strategies, we identify three consumption cases: on-demand instance, reserved or spot blocks, and spot instance. We also consider that although our BPs could be subject to temporal dependency and absolute temporal constraints, we only look into duration constraints. For more details about how we handled other constraints, readers are referred to [2].

- **On-demand instance:** When a is defined as ({Duration(a, $MinD_a$, $MaxD_a$)}, r, on-demand), a is transformed into two timed automata, TA_{ad} and TA_{rd}. On the one hand, TA_{ad}, represents the activity's timed automata and consists of three locations: a_{Ready}, $a_{Working}$, and a_{Finish}. a_{Ready} means that a is ready for execution, $a_{Working}$ means that a is running and consumes r as an on-demand instance, and a_{Finish} means that a has been successfully executed allowing to free r. The transition from a_{Ready} to $a_{Working}$ initializes a clock t to zero. And, the transition from $a_{Working}$ to a_{Finish} takes a guard to control the activity's temporal duration. Formally, , TA_{ad} is a tuple where L={a_{Ready}, $a_{Working}$, a_{Finish}}, l_0=a_{Ready}, X=t, I(a_{Ready})=∅, I($a_{Working}$)={$t \leq MaxD_a$}, I(a_{Finish})=∅, and Tr= $\{(a_{Ready}, start!, t = 0, \varnothing, a_{Working}), (a_{Working}, done?, t \geq MinD_a$ $\&\&t \leq MaxD_a$, $a_{Finish})\}$ (Definition 5).

On the other hand, TA_{rd} consists of three locations: $Idle$, $Inuse$, and $Used$. $Idle$ means that r is available but not-assigned, yet; $Inuse$ means that r is assigned to a and is in-use; and, $Used$ means that r was consumed in the past by a and now is free. Formally, TA_{rd} is a tuple where L={$Idle$, $Inuse$, $Used$}, $X = \varnothing$, l_0=Idle, I(Idle)=∅, I(Inuse)=∅, I(Used)=∅, and Tr={($Idle$, $start?$, ∅, $Inuse$), ($Inuse$, $done!$, ∅, $Used$)} (Definition 5).

For illustration purposes, we consider a_1=({Duration(($a_1, 1, 2$)}, r_1, on-demand) in the "supervision process" BP. a_1 is transformed into two timed automata: TA_{ad} is shown in Figure 2a and TA_{rd} is shown in Figure 2b. a_1 can start consuming r_1 only if the clock t is less than 2 (i.e., defined as an invariant) and will execute successfully only if its temporal duration is met (i.e., defined as a guard).

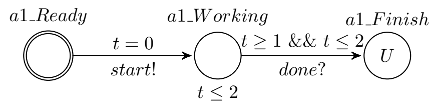


(a) TA_{ad} of activity a_1 (b) TA_{rd} of r_1 as an on-demand instance

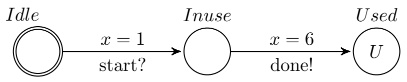
Figure 2: Allocation of r_1 as an on-demand instance cloud-resource to activity a_1

- **Reserved or Spot blocks:** When a is defined as ({Duration(a, $MinD_a$, $MaxD_a$)}, r, reserved) or ({Duration(a, $MinD_a$, $MaxD_a$)}, r, spot blocks) it is transformed into two timed automata, TA_{ar} and TA_{rr}. TA_{ar} and TA_{ad} are similar. Further, TA_{rd} and TA_{rr} have the same formalism in terms of locations, transitions, clock, guard, and invariant. This is due to the fact that a cloud resource consumed as on-demand, reserved or spot blocks, is not subject to any interruption once it becomes consumed. Thus, the resource's timed automata, TA_{rr}, consists of the same three locations as TA_{rd}: *Idle*, *Inuse*, and *Used*. But, they are different in terms of transitions since r can have temporal duration. As a result, we assign a clock x to the transitions between $\{Idle, Inuse\}$ and $\{Inuse, Used\}$ to specify the minimum ($MinAvr$) and maximum ($MaxAvr$) durations of r's temporal availability. Formally, TA_{rr} is a tuple where L=$\{Idle, Inuse, Used\}$, $X = \{x\}$, l_0=Idle, I(Idle)=$\varnothing$, I(Inuse)=$\varnothing$, I(Used)=$\varnothing$, and Tr=$\{(Idle, start?,$ $\{$x= $MinAvr\}, Inuse), (Inuse, done!,$ $\{x = MaxAvr\}, Used)\}$ (Definition 5).

For illustration, we refer to our case study where Figure 3a, same as Figure 2a, presents a_1's timed automata TA_{ar}, and Figure 3b presents r_1's timed automata TA_{rr}. When r_1 takes on *Inuse* state, the clock x is initialized to 1. If it is initialized to 6 that is the maximum duration, then the resource is considered as used and no longer available.



(a) TA_a of activity a_1



(b) TA_{rr} of r_1 as a spot block

Figure 3: Resource allocation with r_1 as spot block to an activity a_1

We implemented our transformation rules using ATLAS Transformation Language (ATL) which is a model-to-model transformation language [22]. Listing 1 is an excerpt of an ATL rule for transforming a resource allocation with spot blocks into a network of timed automata. Figure 6c depicts the generated timed automata. In this listing, Line 2 corresponds to the source element of the BPMN model. Lines 4-7 depict how *Idle* state is created. Similarly to *Idle* state, we create *Inuse* and *Used* states [6]. The way transitions are created is presented in Lines 9-25. Finally, lines 27-33 depict the composition of the full timed-automata of the spot blocks pricing strategy.

Listing 1: Example of a tranformation rule in ATL

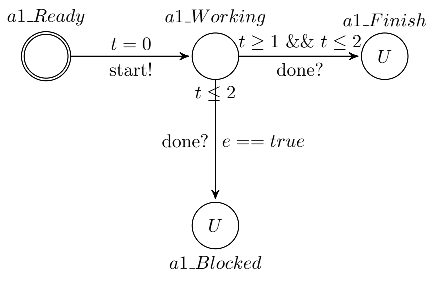
```
 1 rule spotPredDuration {
 2 from spotPred: BPMN!ExtensionAttributeValue (..)
 3 to
 4 idleLocation:uppaal!Location(
 5 id <-'id' + thisModule.getcounterSpotPredefined()
 6 name <-thisModule.NewName('Idle),
 7 initLocation: uppaal!Init  (ref <-idleLocation.id),...
 8
 9 transitionIdleInuse: uppaal!Transition (
10 source <- thisModule.SourceTransition(idleLocation),
11 target <- thisModule.TargetTransition(inUseLocation),
12 label <- thisModule.synchronisation('start?'),
13 label <- spotPred.taskConfig.parameterspotPredDuration ->
14          collect(e |if e.Shared=false then
15                  thisModule.assignmentVM('x=' + e.MinAvR)
16                  else OclUndefined  endif),
17
18 transitionInuseUsed: uppaal!Transition (
19 source <-thisModule.SourceTransition(inUseLocation),
20 target <-thisModule.TargetTransition(usedLocation),
21 label <-thisModule.synchronisation('done!'),
22 label <- spotPred.taskConfig.parameterspotPredDuration ->
23          collect(e |if  e.Shared = false then
24                  thisModule.assignmentVM('x=' + e.MaxAvR)
25                  else OclUndefined endif),

templateSpotPredefined: uppaal!Template (
name <- templateNameSpotPredefined,
location <- idleLocation,...
transition <- transitionIdleInuse,...
init <- initLocation),
templateNameSpotPredefined: uppaal!Name (
value <- 'templateSpotPredifinedDuration')}
```

- **Spot instance:** When a is defined as ({Duration$(a, MinD_a, MaxD_a)$}, r, spot instance), a is transformed into two timed automata TA_{aa} and TA_{ra}. It must be noted that r here can be interrupted while it is under use. On the one hand, TA_{aa} is the activity's timed automata. So, activity a would be blocked. That is why we add a fourth state, $a_{Blocked}$, reached when the resource r becomes interrupted. Defining a Boolean variable e is required to verify if an external event has happened (i.e., another customer proposes a higher bid price and takes resource r). Thereby, transiting from $a_{Working}$ to $a_{Blocked}$ is enabled only if the guard indicating the interruption is true. Formally, TA_{aa} is a tuple where L={a_{Ready}, $a_{Working}$, $a_{Blocked}$, a_{Finish}}, l_0=a_{Ready}, $X = \{x\}$, I(a_{Ready})=∅, I($a_{Working}$)={x $\leq MaxD_a$}, I($a_{Blocked}$)=∅, I(a_{Finish}=∅, Tr={(a_{Ready}, *start*!, $x = 0$, ∅, $a_{Working}$), ($a_{Working}$, *done*?, $e == true$, $a_{Blocked}$), ($a_{Working}$, *done*?, $x \geq MinD_a$ && $x \leq MaxD_a$, a_{Finish})} (Definition 5).

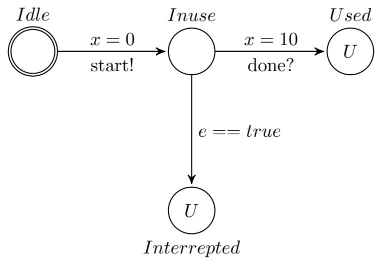
  On the other hand, TA_{ra} is composed of four states: *Idle*, *Inuse*, *Used*, and *Interrupted*. If the spot price becomes greater than the bid price, r will be interrupted. In this case, the transition from *Inuse* to *Interrupted* is enabled. So we use a Boolean variable e to control the interruption. Moreover, to satisfy the absolute constraint over r, we initialize x to zero and take it as a timed reference to subsequently specify that exactly $FUNET - SUNET$ hours must separate the starting and finishing availability times of the resource. Therefore, to move from *Inuse* to *Used*, we update x based on the difference between $FUNET$ and $SUNET$.

Formally, TA_{ra} is a tuple where L=$\{Idle,\ Inuse, Interrupted, Used\}$, l_0=Idle, $X = \{x\}$, I(Idle)=∅, I(Inuse)=∅, I(Interrupted)=∅, I(Used)=∅, and Tr=$\{(Idle, start?, \{$x= 0$\}, Inuse), (Inuse, done!,\ \{x = FUNET - SUNET\},\ Used),\ (Inuse,\ e == true,\ \varnothing,\ Interrupted)\}$ (Definition 5).

For illustration purposes, we consider that a_1=({Duration($(a_1, 1, 2)$)}, r_1, spot instance) in the "supervision process" BP. Figure 4 presents the transformation output of a_1. Figure 4a is different from Figure 3a; $a1_{Blocked}$ location is reached when r_1 is interrupted (i.e., defined as a guard). The timed automata of r_1 is illustrated in Figure 4b. When r_1 is in state *Inuse* the clock x, taken as a time reference, is initialized to zero. Then, it is initialized to 10 if its finish availability time is reached. The Boolean variable e is "true" to indicate that resource r_1 is interrupted (i.e., defined as a guard).



(a) TA_{aa} of a activity a_1



(b) TA_{ra} of r_1 as a spot instance

Figure 4: Allocation of r_1 as a spot instance with an interruption risk to activity a_1

Our transformation rules are correct-by-construction since they are neither ambiguous nor conflicting. The generated network of timed automata that the transformation produces are well-formed. These automata networks are consistent with timed-automata meta-model. In addition, the transformation is complete since each element (e.g., activity, gateway, and event) in a time-constrained BP, as a source model, has a corresponding element (e.g., state, transition, and guard) in the timed automata, as a target model.

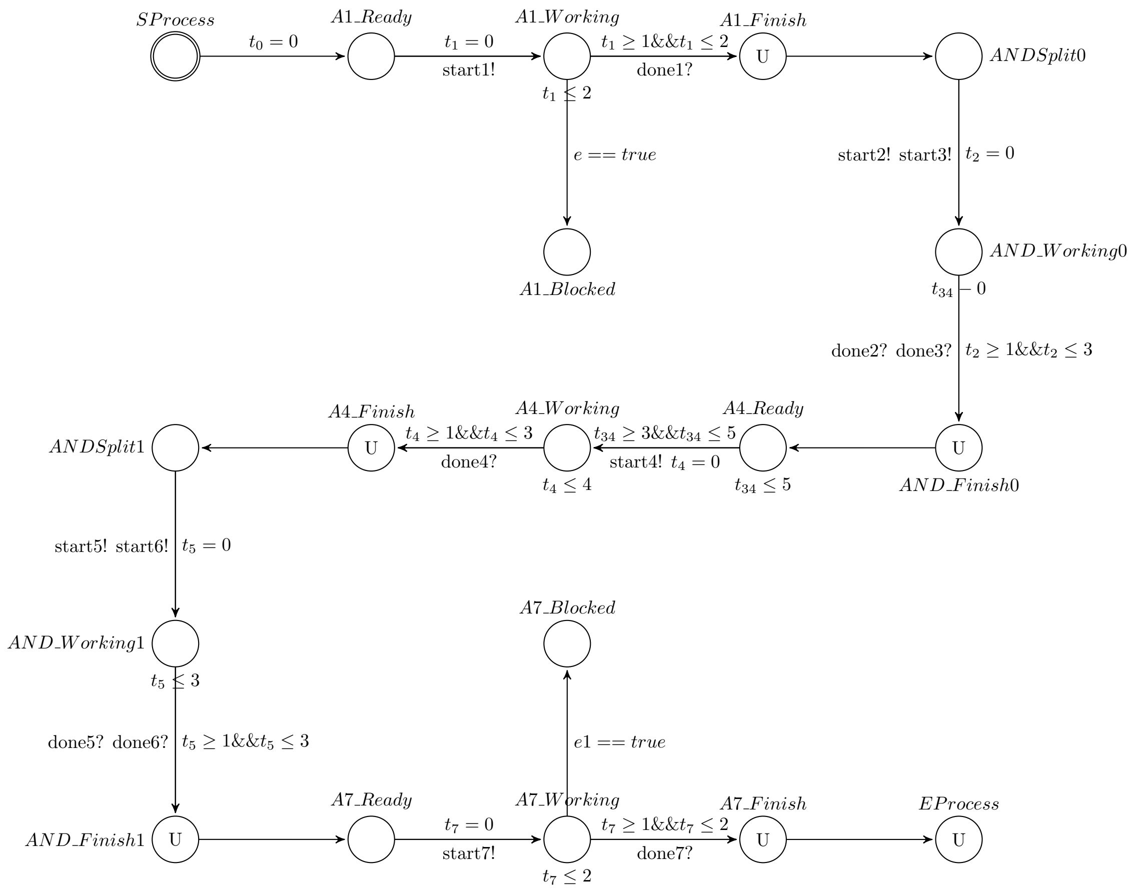
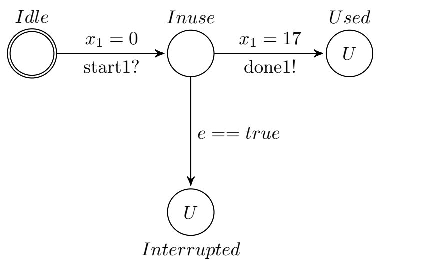


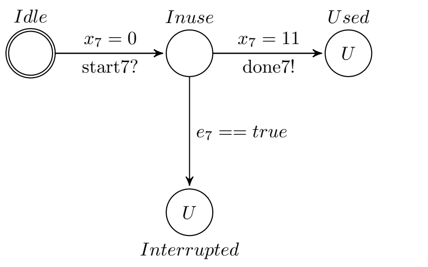
Figure 5: Process activities timed automata

4.2. Correctness analysis

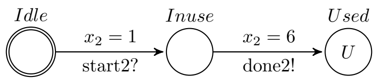
Figures 5 and 6 illustrate the generated network of timed automata associated with “supervision process” BP. We assume that a_1 consumes r_1 as a spot instance, a_2 consumes r_2 as a spot blocks, a_3 consumes r_1 as on-demand instance, a_4 consumes r_5 as an on-demand instance, a_5 consumes r_1 as a spot blocks, a_6 consumes r_3 as a spot instance, and a_7 consumes r_3 as an on-demand instance. We submit the network of timed automata to UPPAAL to verify first, the matching between temporal constraints of both activities and cloud resources and second, the satisfaction of some properties including: liveness,



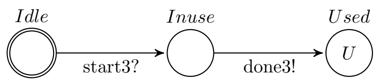
(a) Timed automata of r_1 as a spot instance



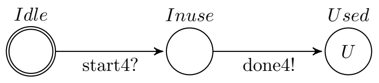
(b) Timed automata of r_3 as a spot instance



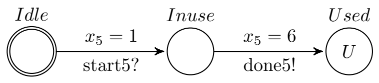
(c) Timed automata of r_2 as a spot blocks



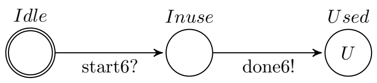
(d) Timed automata of r_1 as an on-demand instance



(e) Timed automata of r_5 as an on-demand instance



(f) Timed automata of r_1 as a spot blocks instance



(g) Timed automata of r_3 as an on-demand instance

Figure 6: Example of a BP transformation into timed automata

free of deadlock, and deadline.

1. Liveness: a specific condition is guaranteed to hold eventually, i.e., "something good will happen (eventually)" [23]. Let us consider that a consumes a spot instance. We say that if a is in state working and the consumed resource is still temporally available then a will eventually finish its execution successfully. Thus, we verify that: a will eventually reach the state finish successfully. With respect to the process in Section 3, if a_2 still consumes r_2 that is still available then eventually a_2 will finish successfully its execution. Thus, the liveness property is satisfied.

2. Free of deadlock: "the system never ends up in a state where it cannot perform any action" [24]. So, we verify that the system is deadlock free

expressed using a special state formula proposed by the model checker UPPAAL.

3. Deadline: it expresses the set of states where a corresponding action is expected to be executed without delay. With respect to “supervision process” BP, its duration is 23 hours, so it should complete in this duration so that the deadline is met. Otherwise, the model checker UPPAAL proposes a counterexample.

5. Cost optimization of BP deployment cost

We discuss how to optimize the cost of BP deployment in cloud resources. To this end, we use Binary Linear Program (BLP) to define both an objective function and constraints that would guide the optimization. BLP is known for its simplicity, flexibility, and extensive modeling capability [25].

5.1. Input and decision variables

To begin with, we assume the availability of several cloud resources that satisfy activities' requests of these resources. BLP takes as an input variable a set of correct allocation options ca that were previously verified in Section 4. Thanks to this verification the BLP's search space and resolution time are reduced [25]. The following are BLP's inputs:

- A set of process activities A, a set of activities' requests of resources $Req_A = \{req_{a_q} : \ a_q \in A\}$, and a set of activities' temporal constraints T_A=$\{temp_{a_q} : a_q \in A\}$;
- A set of cloud providers $Pr = \{pr_{ij}, \forall j \in \{1, \cdots, p\}\}$ and a set of pricing strategies $St_{ij} = \{st_{ijk}, \forall k \in \{1, \cdots, s\}\}$ per provider pr_{ij}.
- A set of resources R' that are correctly allocated using our formal verification approach. $R' = \{r_c, \forall c \in \{1, \cdots, ca\}\}$ where $R' \subseteq R$.

Equation 2 is the decision variable that assigns a suitable cloud resource, cloud provider, and pricing strategy to an activity.

$$X_{ijkq} = \begin{cases} 1 & \text{if activity } a_q \text{ consumes resource } r_i \in R' \text{ that} \\ & \text{provider } pr_{ij} \text{ offers according to strategy } k; \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

5.2. Problem statement

To optimize cost, the model objective function in Equation (3) selects the cloud resources that satisfy activities' requests of resources, the cloud providers of these resources, and the suitable pricing strategies that satisfy these activities' temporal constraints.

$MinC$ is the total execution cost that is the sum of all resources allocated to a process's activities. To compute this cost, we multiply an activity's execution time d_q=$MaxD_a$ by the resource r_i's utilization price (c_{ijk}) that is reported in the strategy k related to the provider pr_{ij} (Equation 3).

$$MinC = \sum_{q=1}^{|A|} \sum_{i=1}^{|R'|} \sum_{j=1}^{p} \sum_{k=1}^{s} d_q c_{ijk} X_{ijkq} \quad (3)$$

In the following, we present a set of linear constraints which should be consistently satisfied to ensure the successful deployment of a BP's activities on cloud resources. A cloud resources allocation refers to a BP's activities whose execution requirements are of types memory and processing. Moreover, both activities and cloud resources are subject to temporal constraints. Finally, we assume that a cloud resource is not shareable and an activity consumes one cloud resource, only.

1. *Execution constraints on activities*: Equations 4 and 5 ensure that resources' capacities meet activities' processing and memory requirements.

$$\sum_{i=1}^{|R'|} \sum_{j=1}^{p} \sum_{k=1}^{s} min(RAM_i) X_{ijkq} \quad \geq \quad RAM_{a_q}, \forall q \quad \in \quad \{1, \cdots, r\} \quad (4)$$

$$\sum_{i=1}^{|R'|}\sum_{j=1}^{p}\sum_{k=1}^{s} min(vCPU_i)X_{ijkq} \quad \geq \quad vCPU_{a_q}, \forall q \quad \in \quad \{1,\cdots,r\} \quad (5)$$

In the case study, activity a_4 requires 28GB and 8 in terms of RAM and $vCPU$, respectively. Therefore, the processing and memory capacities of the selected resource r_i should be equal or greater to RAM_{a_4} and $vCPU_{a_4}$, respectively.

2. *Temporal constraints on activities:* Equation 6 ensures that each activity's start time (S_{a_q}) is after the maximum end-time (F_{a_o}) of all this activity's predecessors.

$$max(F_{a_o}) + TD(FS, a_q, a_o, du, du) \leq S_{a_q} : \forall a_q, a_o \in A \text{ and } o < q \text{ and } D_{max} = du \quad (6)$$

In the case study, a_5 and a_6 are a_7's predecessors; a_7's start time is after their respective end times. Consequently, S_{a_7} must be equal or greater than the maximum value between the respective end times of a_5 and a_6.

3. *Temporal constraints on pricing strategies:* Equations 7 and 8 specify the time interval allowed to allocate resource r_i ($[MinAvr_i, MaxAvr_i]$) to an activity a_q. Equations 9 and 10 ensure that r_i is available from the start until the end time of this activity a_q.

$$\sum_{i=1}^{|R'|}\sum_{j=1}^{p}\sum_{k=1}^{s} MinAvr_i X_{ijkq} \quad \geq \quad MinD_{a_q}, \forall q \quad \in \quad \{1,\cdots,r\} \quad (7)$$

$$\sum_{i=1}^{|R'|}\sum_{j=1}^{p}\sum_{k=1}^{s} MaxAvr_i X_{ijkq} \quad \geq \quad MaxD_{a_q}, \forall q \quad \in \quad \{1,\cdots,r\} \quad (8)$$

$$\sum_{i=1}^{|R'|}\sum_{j=1}^{p}\sum_{k=1}^{s} SUNET(r_i) X_{ijkq} \quad \leq \quad S_{a_q}, \forall q \quad \in \quad \{1,\cdots,r\} \quad (9)$$

$$\sum_{i=1}^{|R'|}\sum_{j=1}^{p}\sum_{k=1}^{s} FUNET(r_i)X_{ijkq} \quad \geq \quad F_{a_q}, \forall q \quad \in \quad \{1,\cdots,r\} \tag{10}$$

Pricing strategies' temporal constraints should satisfy activities' temporal constraints (Equations 7-10). For instance, $MinD_{a_5}$=1 hour and $MaxD_{a_5}$=2 hours are minimum and maximum duration, respectively. If r_i is assigned to a_5 as a spot blocks, then its temporal duration should be greater than d_5, i.e., $MinAvr_i \geq d_5$ and $MaxAvr_i \geq d_5$. However, if r_i is allocated as a spot instance to a_5, then, $SUNET(r_i) \leq S_{a_5}$ and $FUNET(r_i) \geq F_{a_5}$.

4. *Constraint interruption:* To avoid paying the penalty price of a critical activity (i.e., such activity should properly managed to avoid failure) should not consume a resource with an interruption risk (str_k=1) as per Equation (11).

$$\sum_{i=1}^{|R'|}\sum_{j=1}^{p}\sum_{k=1}^{s} X_{ijkq}str_k \;=\; 0 \;:\; \forall a_q \;\in\; A, p_q \;>\; 0 \text{ and } str_k \;=\; 1 \tag{11}$$

In the case study some activities like a_4 are subject to financial penalties, so a_4 should not consume r_i as spot instance (Equation 11).

5. *Assignment constraint:* Equation 12 ensures that a cloud resource has one pricing strategy and is consumed by one activity at the time.

$$\sum_{q=1}^{|R'|} X_{ijkq} = 1 : \forall i \in \{1,\cdots,n\}, \forall j \in \{1,\cdots,p\}, \forall k \in \{1,\cdots,s\} \tag{12}$$

In the case study, a_2 consumes one resource r_2 that the cloud provider pr_{21}=Amazon offers using the pricing strategy st_{213}=spot block and ensures that it is allocated to a_2, only. As a result, X_{2132}=1.

6. *Binary constraint:* to ensure that our linear program is binary, we impose

that the decision variable should be either 0 or 1 (Equation 13).

$$X_{ijkq} \in \{0,1\}, \quad \forall q \in \{1,\cdots,r\}, i \in \{1,\cdots,n\}, j \in \{1,\cdots,p\}, \\ k \in \{1,\cdots,s\} \quad (13)$$

6. Evaluation

We present the technical doability of our approach for ensuring the correctness and optimization of time-aware cloud resources allocation to BPs. This doability started with model checking to verify some Computation Tree Logic (CTL) properties that support matching temporal constraints of both activities and cloud resources. Then, we analyzed the impact of verification on the optimization approach in terms of objective function and response time. Finally, comparing our results to those of CloudSim simulator took place. Due to lack of real datasets that could be used for optimizing the BPs deployment cost, we randomly generated a simulated dataset defined from the ranges presented in Table 4. The different experiments were conducted on a Windows 10, 64-bit Intel Core 2.3 GHz CPU, and 6 GB RAM laptop.

Table 4: Data input ranges

Information	**Type**	**Range**
Number of providers	integer	2
Number of Amazon strategies	integer	$[1, \cdots, 4]$
Number of Microsoft strategies	integer	1
Number of vCPU	integer	$[2, \cdots, 10]$
Amount of RAM (GB)	double	$[15, \cdots, 30]$
Compute price	double	$[0.01\$, \cdots, 0.532\$]$
Requirement in vCPU	integer	$[2, \cdots, 10]$
Requirement in RAM (GB)	double	$[15, \cdots, 30]$
Number of activities	integer	$[5, \cdots, 1000]$
Activities' duration	integer	$[1, \cdots, 5]$
Penalty cost	double	$[0\$, \cdots, 1\$]$

6.1. Checking CTL properties

Model checking is a widely used technique to verify the BP against a wide range of temporal constraints [3]. Our UPPAAL-based verification targets the matching of the temporal constraints of activities and cloud resources. A BP and cloud resource allocation are modeled as a network of timed automata of ProcessActivities (activity-timed automata) and Resource (resource-timed automata), respectively. Assuming a process automata such as the one in Figure 5, we use a rich set of CTL formulae to verify the satisfaction of different properties such as:

1. **Liveness:** *E<>(ProcessActivities.ANDWorking1 and* r_2*.Inuse* $\rightarrow$ *ProcessActivities.ANDFinish1):* if a_2 is consuming r_2 which is still available then eventually a_2 will finish successfully its execution.
2. **Deadlock:** *A[] not deadlock:* the BP is free of deadlock.
3. **Deadline**: *A[](ProcessActivities.EProcess imply t0<=23):* the BP should reach the EProcess state before 23 hours to ensure that its deadline is met.

UPPAAL takes inputs (i) an activity-timed automata (Figure 5), (ii) a resource-timed automata (Figure 6), and (iii) a formulae to formally verify the correctness of the cloud resources allocation. The outcomes show that the corresponding resource allocation is not correct, i.e., the BP is not safe, not deadlock free, and does not meet its deadline. This helps the BP designers to detect temporal violations due to the mismatch between activities' and cloud resources' temporal constraints.

6.2. Impact of verification on optimization

We note that linear programming optimization problems are NP- hard [26]. So they require high computational efforts to find out an optimal and even a feasible solution for large size problems. For that, it is often more important to reduce the search space to avoid waiting for a long time to obtain the optimal solution. To this end, to deal with more complex and large BPs, before moving to the optimization step, we check in our verification approach the temporal correctness of time-aware cloud resource allocation in BPs. Then, we take as inputs for our BLP: a BP and only the set of cloud resources and pricing strategies that ensure correct resource allocations. In this manner, the size of the optimization problem is reduced and so the response time of our BLP is reduced. Consequently, our BLP may converge in a short time to an optimal solution [27]. For that, we vary the optimization problem size to compare between the values of response time and objective function when the inputs of our BLP are a BP, and (i) a set of cloud resources and their pricing strategies (BLP without verification), or (ii) a set of possible allocation options that are verified as per Section 4 (BLP with verification).

In Figure 7 and Figure 8, we respectively present the response time's and objective function's values of our BLP. On the one hand, as expected, the results show that BLP's computation time has low values (about 30 seconds) in case of BPs with a small number of activities (under 200) compared to values (high number of hours) in case of BPs with a large number of activities. We also note that taking as input correct allocations, in both cases, helps to converge in a limited time (under 1 hour) to optimal solutions thanks to a restricted search space. More precisely, an organization can reduce up to 85% in response time to obtain the optimal solution. This observation may support the hypothesis that minimizing the problem size can lead to minimizing the waiting time to reach an optimal solution.

On the other hand, the results revealed that the objective function values are always high if we take as input correct allocations (verification step). This is due to a restricted search space. But, if we take as input all cloud resources (without verification step) we notice that the objective function values are less thanks to the large variety of cloud resources proposed in various pricing strategies. Namely, an organization can save up to 8% in BP deployment cost (i.e., objective function) when enlarging the search space (without verification step). In general, therefore, it seems that the restriction of the search space may cause the raise of the objective function value.

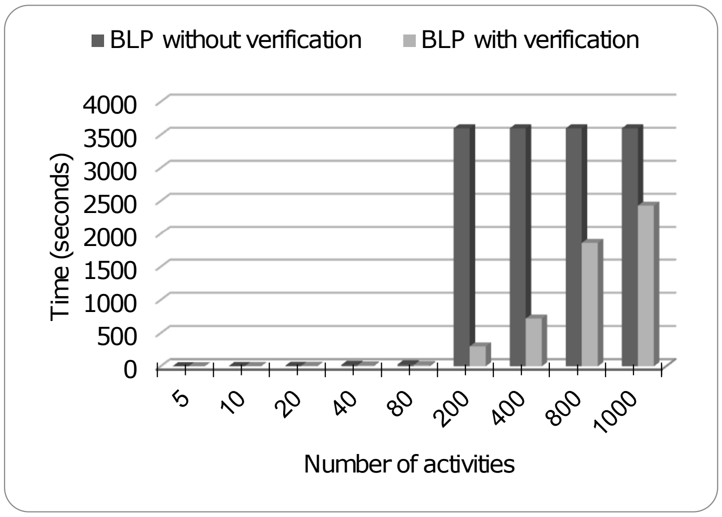


Figure 7: Impact of inputs on the response time

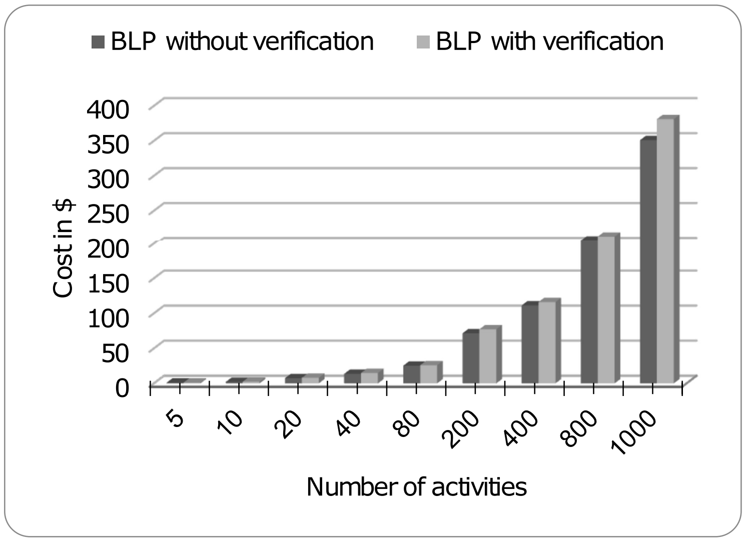


Figure 8: Impact of inputs on the objective function

The results provided from BLP show the effectiveness of our solution. In
590 fact, taken together, these results suggest that our approach helps the designers
to optimize costs of BP deployment in cloud resources. Even in case of space
restriction (with verification) for BPs with a large number of activities, the rate

of increase of the objective function is quite small (up to 8%) compared to the rate of decrease of the response time (up to 85%).

The number of correct allocations *ca* is one of the factors that can impact both: the BP deployment cost and our BLP's response time. Thus, we vary this number *ca* to analyze the values of the response time and the objective function of our BLP. So, we take as inputs: (i) a BP of 200 activities and (ii) a set of correct allocations *ca*. The results are reported in Figure 9 and Figure 10. As shown in Figure 9, the BP deployment cost is always less expensive when *ca* is higher while the search space is larger and so there are more choices. In contrast, from Figure 10, it can be seen that following the increase of *ca*, a significant increase in the response time was recorded. Mainly, the high value of *ca* helps to reduce the objective function but it raises the response time of our BLP. It can therefore be assumed that the number of correct allocations has a significant impact on BP deployment cost and BLP's response time.

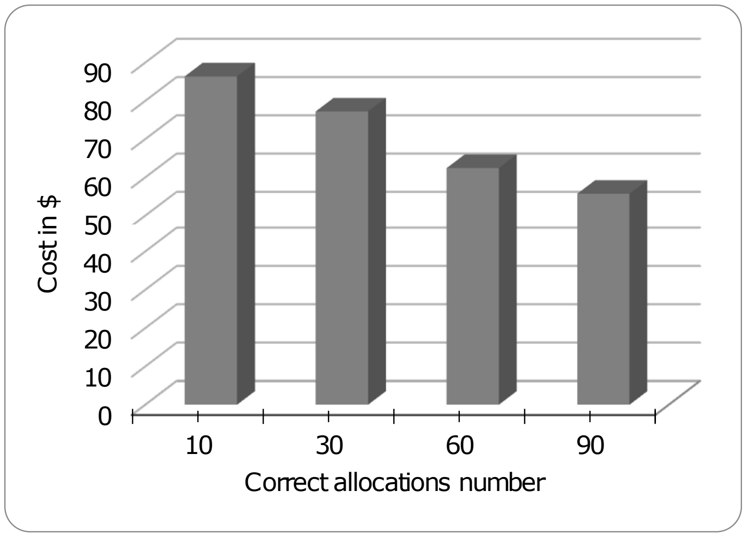


Figure 9: Impact of the number of correct allocations on the objective function

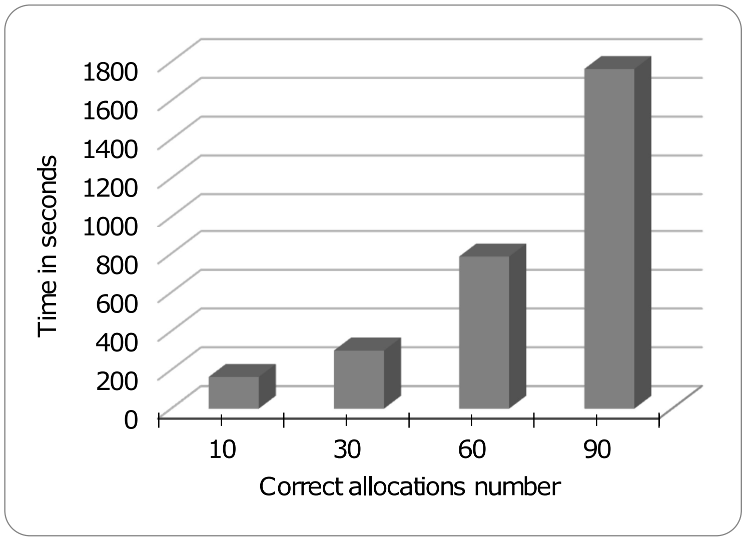


Figure 10: Impact of correct allocations'number on response time

6.3. Simulation Results

The price of cloud resources is variable and depends on temporal perspective such as spot instance. So, configuring a real cloud environment using a wrong estimation can lead to waste of efforts, time, and money. As a result, to ensure that the BP is successfully executed organizations would pay extra fees. To overcome this challenge, researchers often rely on simulation tools to model the mechanisms and evaluate their outputs [28]. That is why, the BP designer can use a cloud simulator in order to simulate a BP deployment in cloud resources and to get its real cost before deploying or even purchasing these resources from cloud providers. We mention that various simulators are suggested to analyze and simulate cloud resource allocation such as CPEE [29], GreenCloud [30], CloudExp [31], CloudSim [32], iCanCloud [33], TeachCloud [34], and many others. Unlike the existence of various simulators, only CloudSim is able to offer a clear isolation of the different multi-layer service abstractions (SaaS, PaaS, and IaaS) requested by cloud computing environment. Besides, because of its extensible behavior, CloudSim allows to seamlessly model, and simulate the emerging cloud infrastructure and application services. It is also open source, developed using java programming language, and does not require a lot of effort and time to implement cloud-based applications. Consequently, we are of the opinion that CloudSim is the most appropriate simulator to use in our experiments.

With the aim of validating the objective function's values (cost) of our approach, we relied on our previous work [35] that extends CloudSim to support AWS pricing strategies and the simulation of a time-constrained BP deployed in cloud resources. Thereby, we show through the experiments that the difference between the cost value of our approach and the cost computed using CloudSim simulator is "limited". Towards this end, we take as input a unified description model specified as an XML document composed of BP activities, temporal constraints, cloud resources, and pricing strategies selected using our BLP model. Thus, based on cloud providers' APIs, CloudSim extension gives the real cost of BP deployment cost.

Figure 11 and Figure 12 illustrate the experimental results. We note the limited difference between the simulator computed cost and the cost value of our approach. Besides, to better compare that difference in various cases (with and without verification) we use Equation (14). The latter computes based on simulator's cost ob_{simul} and our approach's cost ob_{ap} the percentage per value. We note that per values are around 13% in case of BPs with a large number of activities. But, overall, it is under 10%. Whereas, it is null if the number of activities is small. These findings suggest that our approach is more efficient especially if the number of a BP activities is small (under 200) and continues to provide good solutions even in case of BPs with a large number of activities, thanks to our verification approach.

$$per = |1 - \frac{ob_{ap}}{ob_{simul}}| \times 100 \tag{14}$$

7. Related work

In this section, we discuss some works related to formal verification (Section 7.1) and optimization (Section 7.2) of BPM in the context of cloud.

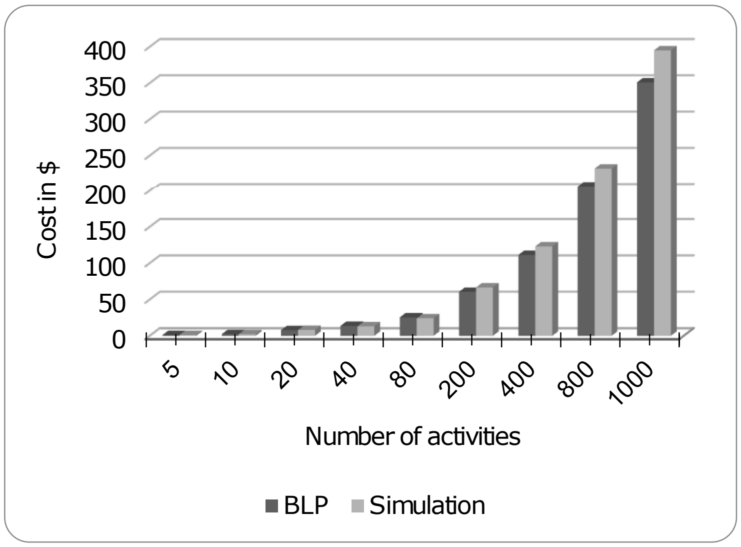


Figure 11: Comparing process cost without verification

7.1. Works on formal verification

Table 5 is a summary of some works that examine cloud resource allocation to time-constrained BP. We classified these works using different criteria such as process temporal constraints, resource, formal language, and pricing strategies. 655 "+" refers to in the scope criteria and "-" refers to out of the scope criterion.

Table 5: Summary of the literature study of formal verification

Work	**Process temporal constraint**	**Resource**	**Formal Language**
[36, 37]	-	+	RAL
[1, 38]	-	+	Event-B
[39]	-	+	Coloured Petri Net
[2, 3, 13]	+	-	Timed automata
[40]	+	-	Time petri net algorithm
[41]	+	+	Timed automata
[42]	+	+	fUML
[43]	+	+	Time workflow net
[44, 45, 46, 47]	+	+	Algorithm
Our approach	+	+	Timed automata

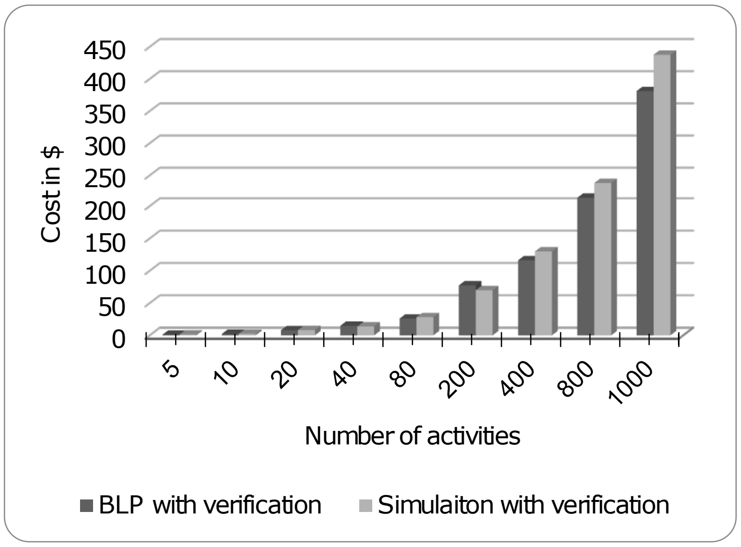


Figure 12: Comparing process cost with verification

A graphical notation was proposed in [36, 37] for assigning human resources to BP activities, the so-called Resource Assignment Language Graph (RALph). The latter has formal semantics provided by a transformation step to Resource Assignment Language (RAL) to automate the analysis of the BP resource perspective [36]. Nevertheless, in both works, process temporal constraints and cloud resources are disregarded.

Boubaker et al. [1] propose a formal specification of resource perspective to guarantee a correct and optimal cloud resource allocation to BPs. Concretely, using the Event-B formal specification, they verify the consistency of cloud resource allocation for process modeling at design time taking into account users' demands and resources' properties. In [38], the same authors verify also the correctness of cloud resource allocation in BPs considering properties such as elasticity and shareability. Garfatta et al. [39] rely on Coloured Petri net formalism to model formally the cloud resource perspective in BP taken into consideration their properties such as vertical/horizontal elasticity. So, the BP designer can model correct resource allocations in BPs to avoid runtime errors. In contrast to the above, we propose rules to transform BPMN models into a network of timed automata models to verify the matching between both activities' temporal constraints and cloud resources' temporal availabilities.

Various research works address the issue of formal specification and verification of temporal constraints in BP models. For instance, Cheikhrouhou et al. in [2, 3, 13] proposed rules to transform BPs modeled in BPMN language and enriched with relative and absolute temporal constraints into a formal language (timed automata). Then, using UPPAAL, they verified the satisfaction of some CTL properties in order to verify the temporal consistency of BPs. Furthermore, Huai et al. [40] verified BPMN models using Timed Petri Nets by analyzing process structure such as deadlock and testing if there is a conflict between model temporal constraints. However, neither (cloud) resource representation nor pricing strategies are studied.

We mention that several works deal with formal verification of resource allocation and activities temporal constraints. In fact, Watahiki et al. [41] extended BPMN to support temporal, concurrency, and resource constraints. Moreover, they generated automatically from BPMN extended models, timed automata models based on a set of transformation rules. The aim is to check deadlocks and bottlenecks. However, Watahiki et al. [41] did not take into account temporal dependency. Besides, Du et al. [43] concentrated on the problem of specification and verification of process temporal constraints. Arévalo et al. [44] focused on activities' temporal constraints and resources. As well, they proposed a framework to derive BPMN models from legacy systems focusing on time and resource perspectives. Compared to our work, authors consider neither advanced temporal constraints, namely the relative and absolute ones, nor cloud resources and pricing strategies. Fakhfakh et al. [48] focused on cloud resource allocation in the context of dynamic workflows. They propose an Event-B model to verify the correctness of dynamic changes. The solution checks properties related to control flow perspective and the matching between activities and cloud resources according to a set of constraints. Contrarily, we take into consideration that each cloud resource in a BP has an assignment interval representing its temporal availability that is associated with some pricing strategies.

Massive parallel business workflows running in the cloud are prone to temporal violations (namely intermediate runtime delays) due to various reasons such as service performance fluctuation and resource conflicts [45]. Thus, authors in [45] presented a propagation-aware temporal verification strategy for parallel business cloud workflows. Further, Luo et al. [46] developed the idea of "adaptiveness" in their design strategy in order to detect temporal violations and to achieve on-time completion of time-constrained business cloud workflows. So, they presented an adaptive temporal checkpoint selection strategy. Besides, they proposed a strategy to handle temporal violation. Besides, Wang et al. [47] proposed a new sliding-window based checkpoint selection strategy for detecting temporal violations. Indeed, the strategy selects the next observation time interval based on the overall temporal consistency state at last checkpoint. However, they did not consider that cloud resources have limited temporal availabilities.

As can be seen from Table 5, the majority of the cited research works focus only on one perspective: time or resource. Moreover, the works combining both perspectives, overall, do not consider that cloud resources have limited temporal availabilities. We observe also, that, in general, authors use formal methods such as timed automata, and petri nets for formal verification reason. Compared to the aforementioned works, we handle advanced temporal constraints on activities and cloud resources' pricing strategies. Next, we transform BPMN models into timed automata models in order to check BPs' time-constraint behaviors such as liveness.

7.2. Works on optimization

Table 6 is a summary of some works that optimize cloud resource allocation cost. We classified these works using different criteria such as process temporal constraints, resource type, resource constraint, etc. "+" refers to in the scope criteria and "-" refers to out of the scope criterion. "+/-" means that the criterion is partially studied.

Afilal et al. [49] proposed an approach for an optimal human resource allocation using MIP model under a set of constraints such as employee availabil-

Table 6: Summary of the literature study of cloud resource allocation optimization

Work	**Process temporal constraint**	**Resource**	**Resource constraint**	**Pricing strategies**	**Optimization technique**
[49]	-	Human	Temporal availability	-	MIP
[50]	-	Cloud	-	-	2 Algorithms
[51]	-	Cloud	-	-	Linear Program
[52]	-	Cloud	-	-	3 Algorithms
[53]	-	Cloud	Various constraints	-on-demand -reserved	-Stochastic integer programming -Algorithm
[54]	-	Cloud	Performance Number of VMs	-on-demand -reserved	Histogram Heuristic
[55]	-	Cloud	QoS constraints Number of VMs	-on-demand -reserved -dynamic	Linear Program Heuristic
[56]	+/-	Cloud	Non shareable	-on-demand -reserved	Stochastic integer programming
[57]	+/-	Cloud	-	-on-demand -reserved	-MIP & Algorithm
[58, 59]	+/-	Cloud	-	-on-demand	Algorithms
[60]	+/-	Cloud	Shareable	-on-demand -reserved	Algorithms
[61]	+/-	Cloud	QoS constraints	-	Meta-Heuristic Algorithms
[62, 63]	+/-	Cloud	-	on-demand	Genetic Algorithms
[64]	+	Cloud	Non shareable	-	Linear Program Algorithm
Our approach	+	Cloud	Temporal Availability	-on-demand -reserved -spot	-Linear Program Heuristic

ity. However, authors considered neither activities' temporal constraints nor cloud resources.

Different research works are proposed in the context of optimal cloud resource allocation cost. For instance, Wang et al. [50] proposed two distributed algorithms to maximize revenues and minimize electricity costs for data centers. Hoenisch et al.[51] proposed a MIP model to optimize the deployment cost of elastic BP under a set of requirements such as the data transfer communication. Moreover, Goettelmann et al. [52] worked on optimizing the cost of deploying BPs into different public clouds. For that, they proposed three algorithms to reduce the cost while considering the data transfer time and resource cost. Though, in our work, we optimize the BP deployment cost while considering different pricing strategies, various cloud providers, and the process constraints (time, capacity, penalty).

A considerable amount of surveys have been presented to extensively review and profoundly study the relevant research works in the context of BPs/workflows scheduling or assignment in a cloud environment such as [65, 66, 67]. Moreover, several algorithms have been proposed for workflow scheduling, but most of them fail to incorporate the key features of cloud including: heterogeneous resources, pay-per-usage model, and elasticity along with the QoS requirements. Kaur et al. [62] proposed a hybrid genetic algorithm which uses the Predict Earliest Finish Time (PEFT) to generate a schedule as a seed with the aim to minimize cost while keeping execution time below the given deadline. A good seed helps to accelerate the process of obtaining an optimal solution. The algorithm is simulated on WorkflowSim and is evaluated using various scientific realistic workflows of different sizes. Visheratin et al. [63] dealt with scheduling scientific workflows in heterogeneous cloud-based computational environment. They considered the main features of IaaS providers, such as the wide variety of offered computational services and pay-as-you-go price model with time periods of charge. More specifically, Visheratin et al. [63] proposed an heuristic algorithm, named Levelwise Deadline Distributed Linewise Scheduling (LDD-LS), for scheduling workflows in hard-deadline constrained clouds. After, they combined LDD-LS with the implementation of IC-PCP algorithm to initialize their proposed metaheuristic algorithm called Cloud Deadline Coevolutional Genetic Algorithm (CDCGA). But, both of studies tend to focus only on one pricing strategy (pay-as-you-go).

In literature, recent research works proposed algorithms to reduce the cost of BP/workflow *scheduling* [58, 59] while taking into consideration heterogeneous resources and hybrid cloud environment. Chen et al. [58] focused on the scheduling problem of budget constrained applications on heterogeneous cloud computing systems. To this end, they suggest an algorithm that minimizes the length of the schedule using the budget level. Namely, this algorithm selects the processors that satisfy the budget constraint and reduce the schedule length. Zhou et al. [59] presented two efficient workflow scheduling approaches to optimize the makespan and monetary cost in hybrid clouds. For this reason, they formulated the workflow scheduling problem, first, as a single objective optimization problem for minimizing the execution cost under deadline constraints and second, as a multi-objective workflow scheduling optimization for minimizing simultaneously the cost and makespan of scheduling workflows. In contrast, Saber et al. [61] offered an approach for Iaas providers to optimize the data centers makespan. For this reason, they formulated the problem of VM reassignment in hybrid and decentralized workflow as a multi-objective problem. They proposed also an hybrid algorithm, called H2-D2, that aims to reassign solutions evaluated by different hosting departments (according to their preferences) using a multi-layer architecture and a metaheuristic algorithm. Nevertheless, in our work we provide for the BP designer an approach that aims to optimize the deployment cost of a time-constrained BP in cloud resources proposed by various cloud providers under different pricing strategies.

Other authors propose approaches to optimize the cloud resource allocation cost while considering 2 pricing strategies (on-demand and reserved) [53, 54]. Chaisiri et al. [53] optimized the cost of cloud resources and the processing time. Moreover, Diaz et al. [54] provided two solutions to optimize cloud resource allocation: (i) a new approach that uses histograms of load levels to find the optimal solution and (ii) an approach that gives an approximated solutions by grouping load levels in bins and computing the histogram. In this manner, the problem size is reduced and the allocation strategy is provided in a short time. Whereas, Bellur et al. [55] considered not only on-demand and reserved but also dynamic pricing strategy. They propose an approach to optimize the cost in multi-site mutli-cloud environments. They formalized the multi-site multi-cloud environment problem and they model it as a linear program. Bellur et al. [55] presented a greedy heuristic algorithm and they have proved its effectiveness in reducing the total cost of infrastructure. Though, in our work, we optimize the process cost while taking into account different pricing strategies (on-demand, reserved, spot), various cloud providers, and the BP constraints (time, capacity).

We mention that several works support partially activities' temporal constraints and consider on-demand and reserved pricing strategies [57, 56, 60]. A stochastic integer programming is proposed by Li et al. [56] to optimize the cloud resource allocation cost and execution time. Hu et al. [57] proposed a MIP model to optimize the deployment cost of elastic BP under a set of requirements such as the data transfer communication. Mastelic et al. [60] defined an approach to estimate and reduce the cost of cloud resources allocated to run process. Concretely, they predicted the execution path of the process to assign the cloud resources to activities and select the best strategy for each cloud resource.

Fakhfakh et al. [64] proposed an approach to minimize the cost of the cloud resources consumed to execute dynamic workflows under a set of constraints. Afterwards, they extended their approach to deal with workflow dynamic changes. In contrast to the above, we optimize the BP deployment cost taken into account cloud resources offered by various cloud providers in various pricing strategies and the activity subject to financial penalty price.

As mentioned in the Table 6, advanced temporal constraints for activities requiring resources to be performed are not well studied in the literature. Furthermore, we conclude that most of the cited research works focus especially on cloud resources. Compared to our work, few works consider that we can specify temporal availability constraint over cloud resources based on pricing strategies.

Besides, the majority of the presented works assume that cloud resources are offered only as on-demand and/or reserved instances. However, in this paper, we take resources from various cloud providers under various pricing strategies. In addition, we denote that mathematical models namely mixed integer programming, linear programming, and stochastic integer programming models are, generally, the most used methods to find the optimal resource allocation cost. Finally, to deal with huge number of input data, authors propose to reduce the complexity of the mathematical model and provide an approximative solution.

8. Conclusion

In this paper, we presented an approach for allocating cloud resources to activities of BPs that are subject to time constraints. This allocation needs to be, at run-time, both correct to avoid blockage and optimized to avoid excessive deployment costs. To achieve verification, we developed rules that transform BPMN-based BPs' process models into a network of timed-automata so that proper matching of activities' needs of resources to cloud resources is ensured despite the time constraints. And, to achieve optimization that the number of a BP's activities could impact, we developed a linear programming model that took into account the deployment cost of these activities over cloud resources.

In term of future work, different aspects will be pursued. First, we would like to verify the semantic correctness of the transformation rules: The generated timed automata should preserve semantic properties like termination (i.e., transformations should always lead to a result) and confluence (i.e., result should be unique). Second, we would like to consider all branching types (i.e., ORsplit and XORsplit) that could define process models of BPs. Currently, we only examined ANDsplit branching. Third, we would like to assess the impact of changes of spot instance price on cloud resources allocation at run-time. Related to this impact, it is worth mentioning Amazon new pricing strategy known as *SavingsPlans* [11] that could be added to our future work plan so that we would have covered all Amazon pricing strategies. Finally we would like to consider data transfer fee when verifying and optimizing cloud resources allocation.

References

[1] S. Boubaker, W. Gaaloul, M. Graiet, N. B. Hadj-Alouane, Event-b based approach for verifying cloud resource allocation in business process, in: Proceedings of the IEEE International Conference on Services Computing, IEEE, 2015, pp. 538–545. doi:10.1109/SCC.2015.79.

[2] S. Cheikhrouhou, S. Kallel, N. Guermouche, M. Jmaiel, Enhancing formal specification and verification of temporal constraints in business processes, in: Proceedings of the IEEE international conference on services computing, IEEE, 2014, pp. 701–708. doi:10.1109/SCC.2014.97.

[3] S. Cheikhrouhou, S. Kallel, M. Jmaiel, Toward a verification of time-centric business process models, in: Proceedings of the IEEE 23rd International WETICE Conference, IEEE, 2014, pp. 326–331. doi:10.1109/WETICE.2014.75.

[4] E. Hachicha, W. Gaaloul, Towards resource-aware business process development in the cloud, in: Proceedings of IEEE 29th International Conference on Advanced Information Networking and Applications, IEEE, 2015, pp. 761–768. doi:10.1109/AINA.2015.265.

[5] R. B. Halima, S. Kallel, K. Klai, W. Gaaloul, M. Jmaiel, Formal verification of time-aware cloud resource allocation in business process, in: Proceedings of the OTM Confederated International Conferences On the Move to Meaningful Internet Systems, Springer, 2016, pp. 400–417. doi:10.1007/978-3-319-48472-3_23.

[6] R. Ben Halima, I. Zouaghi, S. Kallel, W. Gaaloul, M. Jmaiel, Formal verification of temporal constraints and allocated cloud resources in business

processes, in: Proceedings of the IEEE 32nd International Conference on Advanced Information Networking and Applications, IEEE, 2018, pp. 952–959. doi:10.1109/AINA.2018.00139.

[7] A. Wall, K. Sandstrom, J. Maki-Turja, C. Norstrom, W. Yi, Verifying temporal constraints on data in multi-rate transactions using timed automata, in: Proceedings Seventh International Conference on Real-Time Computing Systems and Applications, IEEE, 2000, pp. 263–270. doi:10.1109/RTCSA.2000.896400.

[8] R. B. Halima, S. Kallel, W. Gaaloul, M. Jmaiel, Optimal cost for time-aware cloud resource allocation in business process, in: Prceedings of the IEEE International Conference on Services Computing (SCC), IEEE, 2017, pp. 314–321. doi:10.1109/SCC.2017.47.

[9] Online, Microsoft azure, https://azure.microsoft.com/en-us/.

[10] Online, Google cloud, https://cloud.google.com/.

[11] Amazon ec2, https://aws.amazon.com/fr/ec2/pricing/ (2020).

[12] M. Weske, Business process management architectures, in: Business Process Management, Springer, 2012, pp. 333–371. doi:10.1007/978-3-642-28616-2_7.

[13] S. Cheikhrouhou, S. Kallel, N. Guermouche, M. Jmaiel, Toward a time-centric modeling of business processes in bpmn 2.0, in: Proceedings of International Conference on Information Integration and Web-based Applications & Services, ACM, 2013, p. 154. doi:10.1145/2539150.2539182.

[14] S. Cheikhrouhou, S. Kallel, N. Guermouche, M. Jmaiel, The temporal perspective in business process modeling: a survey and research challenges, Service Oriented Computing and Applications 9 (1) (2015) 75–85. doi:10.1007/s11761-014-0170-x.

[15] M. Pesic, M. H. Schonenberg, N. Sidorova, W. M. P. van der Aalst, Constraint-based workflow models: Change made easy, in: R. Meersman, Z. Tari (Eds.), Proceedings of the International Conferences on the Move to Meaningful Internet Systems, Vol. 4803 of Lecture Notes in Computer Science, Springer, 2007, pp. 77–94. doi:10.1007/978-3-540-76848-7_7.

[16] D. Gagné, A. Trudel, Time-bpmn, in: B. Hofreiter, H. Werthner (Eds.), Proceedings of the IEEE Conference on Commerce and Enterprise Computing, IEEE Computer Society, 2009, pp. 361–367. doi:10.1109/CEC.2009.71.

[17] W. Huai, X. Liu, H. Sun, Towards trustworthy composite service through business process model verification, in: Proceedings of the 2010 7th International Conference on Ubiquitous Intelligence and Computing and 7th International Conference on Autonomic & Trusted Computing, IEEE Computer Society, 2010, pp. 422–427. doi:10.1109/UIC-ATC.2010.114.

[18] I. Graja, S. Kallel, N. Guermouche, A. H. Kacem, Time patterns for cyber-physical systems, in: Proceedings of the IEEE Symposium on Computers and Communication, IEEE Computer Society, 2016, pp. 1208–1211. doi:10.1109/ISCC.2016.7543900.

[19] G. Rodriguez-Navas, J. Proenza, Using timed automata for modeling distributed systems with clocks: Challenges and solutions, IEEE Transactions on Software Engineering 39 (6) (2012) 857–868. doi:10.1109/TSE.2012.73.

[20] E. Hachicha, N. Assy, W. Gaaloul, J. Mendling, A configurable resource allocation for multi-tenant process development in the cloud, in: Proceeedings of the 28th International Conference on Advanced Information Systems Engineering, Springer, 2016, pp. 558–574. doi:10.1007/978-3-319-39696-5_34.

[21] Omg. business process model and notation (bpmn) 2.0., http://www.omg.org/spec/BPMN/2.0/.

[22] Atlas transformation language, `https://www.eclipse.org/atl` (2020).

[23] L. Lamport, Proving the correctness of multiprocess programs, IEEE transactions on software engineering 3 (2) (1977) 125–143. `doi:10.1109/TSE.1977.229904`.

[24] W. Fokkink, A. Kakebeen, J. Pang, Adapting the uppaal model of a distributed lift system, in: International Conference on Fundamentals of Software Engineering, Springer, 2007, pp. 81–97. `doi:10.1007/978-3-540-75698-9_6`.

[25] D. Solow, Linear and nonlinear programming, Wiley Encyclopedia of Computer Science and Engineering (2007) 1–5`doi:10.1002/9780470050118.ecse219`.

[26] G. R. Raidl, J. Puchinger, Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization, in: Hybrid metaheuristics, Springer, 2008, pp. 31–62.

[27] M. Laguna, J. Marklund, Business Process Modeling, Simulation and Design, CRC Press, 2013.
URL `https://books.google.be/books?id=SRHSBQAAQBAJ`

[28] W. Long, L. Yuqing, X. Qingxin, Using cloudsim to model and simulate cloud computing environment, in: Proceedings of the 9th International Conference on Computational Intelligence and Security, IEEE, 2013, pp. 323–328. `doi:10.1109/CIS.2013.75`.

[29] J. Mangler, S. Rinderle-Ma, CPEE - cloud process execution engine, in: Proceedings of the BPM Demo Sessions, co-located with the 12th International Conference on Business Process Management, 2014, p. 51.

[30] D. Kliazovich, P. Bouvry, S. U. Khan, Greencloud: a packet-level simulator of energy-aware cloud computing data centers, The Journal of Supercomputing 62 (3) (2012) 1263–1283. `doi:10.1007/s11227-010-0504-1`.

[31] Y. Jararweh, M. Jarrah, M. Kharbutli, Z. Alshara, M. N. Alsaleh, M. Al-Ayyoub, Cloudexp: A comprehensive cloud computing experimental framework, Simulation Modelling Practice and Theory 49 (2014) 180–192. doi:10.1016/j.simpat.2014.09.003.

[32] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and experience 41 (1) (2011) 23–50. doi:10.1002/spe.995.

[33] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, I. M. Lorente, iCanCloud: A flexible and scalable cloud infrastructure simulator, Journal of Grid Computing 10 (1) (2012) 185–209. doi:10.1007/s10723-012-9208-5.

[34] Y. Jararweh, Z. Alshara, M. Jarrah, M. Kharbutli, M. N. Alsaleh, Teachcloud: a cloud computing educational toolkit, International Journal of Cloud Computing 2 (2/3) (2013) 237–257. doi:10.1504/IJCC.2013.055269.

[35] R. B. Halima, S. Kallel, M. A. Nacer, W. Gaaloul, Optimal business process deployment cost in cloud resources, Journal of Supercomputing (2020).doi:10.1007/s11227-020-03316-9.

[36] C. Cabanillas, D. Knuplesch, M. Resinas, M. Reichert, J. Mendling, A. Ruiz-Cortés, Ralph: a graphical notation for resource assignments in business processes, in: Proceedings of the International Conference on Advanced Information Systems Engineering, Springer, 2015, pp. 53–68. doi:10.1007/978-3-319-19069-3_4.

[37] C. Cabanillas, M. Resinas, A. del Río-Ortega, A. Ruiz-Cortés, Specification and automated design-time analysis of the business process human resource perspective, Information Systems 52 (2015) 55–82. doi:10.1016/j.is.2015.03.002.

[38] S. Boubaker, A. Mammar, M. Graiet, W. Gaaloul, Formal verification of cloud resource allocation in business processes using event-b, in: Proceedings of the IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), IEEE, 2016, pp. 746–753. doi:10.1109/AINA.2016.126.

[39] I. Garfatta, K. Klai, M. Graiet, W. Gaaloul, Formal modelling and verification of cloud resource allocation in business processes, in: Proceedings of the Confederated International Conferences On the Move to Meaningful Internet Systems, Springer, 2018, pp. 552–567. doi:10.1007/978-3-030-02610-3_31.

[40] W. Huai, X. Liu, H. Sun, Towards trustworthy composite service through business process model verification, in: Proceedings of the 7th International Conference on Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing, IEEE, 2010, pp. 422–427. doi:10.1109/UIC-ATC.2010.114.

[41] K. Watahiki, F. Ishikawa, K. Hiraishi, Formal verification of business processes with temporal and resource constraints, in: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2011, pp. 1173–1180. doi:10.1109/ICSMC.2011.6083857.

[42] Y. Laurent, R. Bendraou, S. Baarir, M.-P. Gervais, Formalization of FUML: An application to process verification, in: Proceedings of the International Conference on Advanced Information Systems Engineering, Springer, 2014, pp. 347–363. doi:10.1007/978-3-319-07881-6_24.

[43] Y. Du, P. Xiong, Y. Fan, X. Li, Dynamic checking and solution to temporal violations in concurrent workflow processes, IEEE Trans. Systems, Man, and Cybernetics, Part A 41 (6) (2011) 1166–1181. doi:10.1109/TSMCA.2011.2116003.

[44] C. Arévalo Maldonado, I. Ramos Román, M. J. Escalona Cuaresma, Discovering business models for software process management-an approach

for integrating time and resource perspectives from legacy information systems, in: Proceedings of the 17th International Conference on Enterprise Information Systems, SciTePress, 2015, pp. 353–359. doi:10.5220/0005454903530359.

[45] H. Luo, X. Liu, J. Liu, Y. Yang, Propagation-aware temporal verification for parallel business cloud workflows, in: Proceedings of the IEEE International Conference on Web Services, IEEE, 2017, pp. 106–113. doi:10.1109/ICWS.2017.22.

[46] H. Luo, X. Liu, J. Liu, B. Han, Y. Yang, Adaptive temporal verification and violation handling for time-constrained business cloud workflows, in: Proceedings of the International Conference on Service-Oriented Computing, Springer, 2018, pp. 90–99. doi:10.1007/978-3-030-03596-9_6.

[47] Y. Wang, R. Xu, F. Wang, H. Luo, M. Wang, X. Liu, Sliding-window based propagation-aware temporal verification for monitoring parallel cloud business workflows, in: Proceedings of the IEEE 22nd International Conference on Computer Supported Cooperative Work in Design, IEEE, 2018, pp. 449–454. doi:10.1109/CSCWD.2018.8465205.

[48] F. Fakhfakh, H. H. Kacem, A. H. Kacem, F. Fakhfakh, Preserving the correctness of dynamic workflows within a cloud environment, Procedia Computer Science 126 (2018) 1541–1550. doi:10.1016/j.procs.2018.08.127.

[49] M. Afilal, H. Chehade, F. Yalaoui, The human resources assignment with multiple sites problem, International Journal of Modeling and Optimization 5 (2) (2015) 155. doi:10.7763/IJMO.2015.V5.453.

[50] W. Wang, P. Zhang, T. Lan, V. Aggarwal, Datacenter net profit optimization with individual job deadlines, in: Proceedings of the 46th Annual Conference on Information Sciences and Systems, 2012, pp. 1–6. doi:10.1109/CISS.2012.6310925.

[51] P. Hoenisch, C. Hochreiner, D. Schuller, S. Schulte, J. Mendling, S. Dustdar, Cost-efficient scheduling of elastic processes in hybrid clouds, in: Proceedings of the 8th IEEE International Conference on Cloud Computing, IEEE, 2015, pp. 17–24. `doi:10.1109/CLOUD.2015.13`.

[52] E. Goettelmann, W. Fdhila, C. Godart, Partitioning and cloud deployment of composite web services under security constraints, in: Proceedings of the IEEE International Conference on Cloud Engineering, IEEE, 2013, pp. 193–200. `doi:10.1109/IC2E.2013.22`.

[53] S. Chaisiri, B.-S. Lee, D. Niyato, Optimization of resource provisioning cost in cloud computing, IEEE transactions on services Computing 5 (2) (2011) 164–177. `doi:10.1109/TSC.2011.7`.

[54] J. L. Díaz, J. Entrialgo, M. García, J. García, D. F. García, Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing, Future Generation Computer Systems 71 (2017) 129–144. `doi:10.1016/j.future.2017.02.004`.

[55] U. Bellur, A. Malani, N. C. Narendra, Cost optimization in multi-site multi-cloud environments with multiple pricing schemes, in: Proceedings of the 7th IEEE International Conference on Cloud Computing, IEEE, 2014, pp. 689–696. `doi:10.1109/CLOUD.2014.97`.

[56] Q. Li, Y. Guo, Optimization of resource scheduling in cloud computing, in: Proceedings of the 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, IEEE, 2010, pp. 315–320. `doi:10.1109/SYNASC.2010.8`.

[57] M. Hu, J. Luo, B. Veeravalli, Optimal provisioning for scheduling divisible loads with reserved cloud resources, in: Proceedings of the 18th IEEE International Conference on Networks, IEEE, 2012, pp. 204–209. `doi:10.1109/ICON.2012.6506559`.

[58] W. Chen, G. Xie, R. Li, Y. Bai, C. Fan, K. Li, Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems, Future Generation Computer Systems 74 (2017) 1–11. `doi:10.1016/j.future.2017.03.008`.

[59] J. Zhou, T. Wang, P. Cong, P. Lu, T. Wei, M. Chen, Cost and makespan-aware workflow scheduling in hybrid clouds, Journal of Systems Architecture 100. `doi:10.1016/j.sysarc.2019.08.004`.

[60] T. Mastelic, W. Fdhila, I. Brandic, S. Rinderle-Ma, Predicting resource allocation and costs for business processes in the cloud, in: Proceedings of the IEEE world congress on services, IEEE, 2015, pp. 47–54. `doi:10.1109/SERVICES.2015.16`.

[61] T. Saber, J. Thorburn, L. Murphy, A. Ventresque, Vm reassignment in hybrid clouds for large decentralised companies: A multi-objective challenge, Future Generation Computer Systems 79 (2018) 751–764. `doi:10.1016/j.future.2017.06.015`.

[62] G. Kaur, M. Kalra, Deadline constrained scheduling of scientific workflows on cloud using hybrid genetic algorithm, in: Proceedings of the 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence, IEEE, 2017, pp. 276–280. `doi:10.1109/CONFLUENCE.2017.7943162`.

[63] A. A. Visheratin, M. Melnik, D. Nasonov, Workflow scheduling algorithms for hard-deadline constrained cloud environments, Procedia Computer Science 80 (2016) 2098–2106. `doi:10.1016/j.procs.2016.05.529`.

[64] F. Fakhfakh, H. H. Kacem, A. H. Kacem, Dealing with structural changes on provisioning resources for deadline-constrained workflow, The Journal of Supercomputing 73 (7) (2017) 2896–2918. `doi:10.1007/s11227-016-1823-7`.

[65] E. N. Alkhanak, S. P. Lee, S. ur Rehman Khan, Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities, Future Generation Computer Systems 50 (2015) 3–21. doi:10.1016/j.future.2015.01.007.

[66] A. R. Arunarani, D. Manjula, V. Sugumaran, Task scheduling techniques in cloud computing: A literature survey, Future Generation Computer Systems 91 (2019) 407–415. doi:10.1016/j.future.2018.09.014.

[67] L. Thai, B. Varghese, A. Barker, A survey and taxonomy of resource optimisation for executing bag-of-task applications on public clouds, Future Generation Computer Systems 82 (2018) 1–11. doi:10.1016/j.future.2017.11.038.